Popular Computing

The world's only magazine devoted to the art of computing.

October 1980

Volume 8 Number 10

91

This is Factorial 11000

316246240647804772964717834667
733148312352964165418043350226
793444991465058722779760834101
5830002153559819941652...

36705 digits
omitted here

....48453312388261330008827929
056400152870915856310995663618
863843732979810783760835911330
025575841231852468337956093952

and 2748 zeros.

...a world's record

...really high precision calculation

It all started with a tentative solution to Problem 120 (from our issue number 36) which appeared in issue 85. The problem concerned the sequence of non-zero low-order digits of successive factorials. The first 1000 of those digits had been calculated, primarily as an exercise in the use of floating point BASIC.

There then followed much discussion of this sequence of digits with David Ferguson, who established that the sequence does not cycle, but that repeating strings of any length can be found.

We have reproduced the listing of the first 1000 of these digits (taken from page 3 of issue 85), together with the next 2000 digits. Mr. Ferguson noticed that in the first 1000 digits, the sequence starting at the 627th digit repeated the sequence starting at the 2nd digit, so that a repetitive cycle of 625 digits was evident even from the limited data that was given.

(The tables are on pages 4 and 5)

Not to doubt for a moment as emminent a worker as Ferguson, but in the interests of acquiring more data, it seemed expedient to extend the calculations. The program in BASIC would be much too slow; moreover, it could not be extended very far due to space limitations. How about rewriting the whole thing in 6502 machine language?

This is quite feasible. To go to, say, factorial 5000, it would be necessary to operate on at least 1300-digit numbers at any one stage.

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $20.50 per year, or $17.50 if remittance accompanies the order. For Canada and Mexico, add $1.50 per year. For all other countries, add $3.50 per year. Back issues $2.50 each.

The reasoning to arrive at that figure of 1300 digits is quite simple. To get to factorial 5000, concerning ourselves only with the low-order non-zero digits, it is necessary to anticipate how many low-order zeros there can be. In developing the factorials systematically, every time the argument is a multiple of 5, the function gains a low-order zero. Each time the argument is a multiple of 25, the function gains two low-order zeros, and so on. Thus, for factorial 5000, we have:

Multiples of 5:	1000
Multiples of 25:	200
Multiples of 125:	40
Multiples of 625:	8
Multiples of 3125:	1
	1249

So if 1300 digits are carried throughout the calculations, and the low-order zeros are shifted off at the right hand end of the number, the 51 extra digits should provide enough protection to insure that the low-order, non-zero digit is correct.

With the usual (lazy) technique of carrying one decimal digit per word, there is no problem of storage space; 1300 consecutive words of storage are cheap.

Multiplication is always a problem on a machine like the 6502 which lacks a MULTIPLY operation, and even more so when the multiplier can get to 5000 and the word size of 8 bits limits the natural range of numbers to plus or minus 127. So the following scheme was adopted.

The index of the factorial being developed, N, will be contained in 3 words of storage in a base-40 scheme, so that the equivalencies shown in Table T hold.

For each new value of N (say, 1234), there will be formed immediately in storage (again using 3 words for each number) the 9 possible multiples of N. Thus, to form the next value of (N!), it is necessary only to look up the proper multiple in this table for each digit of (N-1)!

Text continues on page 6

24284484666264662648868288682662644484644846224284

48468868222428224286626488682662644484644846224288

86826626444846448462242844846886822242822428662646

62642224288868288682448468868266264448464484622428 6

62642224288868288682448466626422428886828868244846 2

24284484666264662648868288682662644484644846224284

48468868222428224286626466682662644484644846224288

86826626444846448462242866264224288868288682448464

48468868222428224286626422428448466626466264886824

48468868222428224286626444846886822242822428662646

62642224288868288682448464484688682224282242866264 2

24284484666264662648868244846886822242822428662644

48468868222428224286626444846886822242822428662646

62642224288868288682448468868266264448464484622428 6

62642224288868288682448466626422428886828868244846 8

86826626444846448462242822428448466626466264886826

62642224288868288682448462242844846662646626488682 2

24284484666264662648868266264224288868288682448464

48468868222428224286626422428448466626466264886824

48468868222428224286626444846886822242822428662648

86826626444846448462242822428448466626466264886826

62642224288868288682448462242844246662646626488682 2

24284484666264662648868288682662644484644846224282

the low-order, non-zero digits of successive factorials. For the first 1000 factorials, the table at the upper left on the facing page is reproduced from issue 85. The table on this page contains the digits from N = 1001 through N = 2150, 50 digits per line. Triple digits are underlined, to aid in tracing patterns. The table at the lower right on the facing page contains the digits for N = 2151 through N = 3250.

12642242888682886824484644846886822242822428662642
24284484666264662648868244846886822242822428662644
48468868222428224286626488682662644484644846224282
24284484666264662648868266264224288868288682448462
24284484666264662648868222428448466626466264886828
86826626444846448462242822428448466626466264886826
62642242888682886824484622428448466626466264886822
24284484666264662648868222428448466626466264886828
86826626444846448462242844846886822242822428662648
86826626444846448462242888682662644484644846224284
48468868222428224286626466264224288868288682448468
86826626444846448462242866264224288868288682448466
62642242888682886824484666264224288868288682448464
48468868222428224286626422428448466626466264886824
48468868222428224286626444846886822242822428662648
86826626444846448462242822428448466626466264886826
62642242888682886824484622428448466626466264886822
24284484666264662648868288682662644484644846224282
24284484666264662648868266264224288868288682448462
24284484666264662648868222428448466626466264886822

The single, low-order, non-zero digits of the first thousand factorials.

24284484666264662648868266264224288868288682448462
24284484666264662648868222428448466626466264886822
24284484666264662648868288682662644484644846224284
48468868222428224286626488682662644484644846224288
86826626444846448462242844846886822242822428662646
62642242888682886824484688682662644484644846224286
62642242888682886824484666264224288868288682448464
48468868222428224286626466264224288868288682448468
86826626444846448462242866264224288868288682448466
62642242888682886824484622428448466626466264886828
86826626444846448462242844846886822242822428662648
86826626444846448462242888682662644484644846224282
24284484666264662648868288682662644484644846224284
48468868222428224286626488682662644484644846224288
86826626444846448462242888682662644484644846224282
24284484666264662648868266264224288868288682448462
24284484666264662648868222428448466626466264886826
62642242888682886824484644846886822242822428662642
24284484666264662648868244846886822242822428662644
48468868222428224286626422428448466626466264886828
86826626444846448462242844846886822242822428662648
86826626444846448462242888682662644484644846224286

	N (decimal)	N (base 40 scheme--expressed in hexadecimal notation)		
	20	00	00	14
	39	00	00	27
	40	00	01	00
	100	00	02	14
	200	00	05	00
	400	00	0A	00
	1000	00	19	00
	1599	00	27	27
	1600	01	00	00
	5000	03	05	00
(1)	1234	00	1E	22
(2)	2468	01	15	1C
(3)	3702	02	0C	16
(4)	4936	03	03	10
(5)	6170	03	22	0A
(6)	7404	04	19	04
(7)	8638	05	0F	26
(8)	9872	06	06	20
(9)	11106	06	25	1A

For example, suppose we have the following sequence of digits in storage for factorial 1233:

....471953628....

and we are in the process of developing factorial 1234. For each digit involved, there is a Carry to consider, and Carry is also 3 words of storage.

The maximum value that Carry can have is almost nine times the maximum N, or 45000; in the notation we are using, Carry will appear as (03 04 05) for the decimal number 4965. At the start of the cycle that develops a new factorial, Carry is set to (00 00 00).

Then, working from right to left, to replace the digit 8, the algorithm calls for looking up the 8th multiple of N, which is (06 06 20), adding the Carry value at that point, and reducing the result to the same form. The digit corresponding to the unit's digit of that result is stored back where the 8 was, and what is left is the value of Carry to use for calculating the next digit to the left.

Having produced a part of each of the factorials up to factorial 5000, it seemed feasible to go further and use the same techniques to develop complete factorials. The opportunity was at hand to break a world's record, which is always fun.

One should not plunge into such a project without first considering how the result might be validated. The last significant work (reported in our issue number 3) was done by Timothy Croy in 1972, to calculate factorial 10000. Mr. Croy thoughtfully printed out many intermediate results, and these could be used as checks in the new calculation of factorial 11000.

An independent calculation was made, to sum the common logarithms of the numbers from 2 to 11000; this sum (made on an SR-52) was:

39680.4999853

which indicates that factorial 11000 has 39681 digits, the first few of which are 31621... Further, we can calculate, as we did before, the number of low-order zeros, which is 2748.

A further check was furnished by Herman P. Robinson, using Stirling's formula, giving the first 45 digits of the expected result.

There is one more interesting aspect to this problem. We can set up a work space of 39681 words, initialized to zeros except for a 2 in the low-order word, and start producing factorials with N = 3. Clearly, this will be extremely inefficient, inasmuch as the result for the first product is contained in one word, but the straightforward procedure will involve 39680 "multiplications" by zero. What is needed is some scheme to allow the multiplications to proceed only as far as is needed for each value of N. All sorts of schemes suggest themselves, but each one requires elaborate coding to implement it, and that code chews up execution time, too. Fortunately, there is a simple scheme that just happens to apply here.

Since factorial 11000 has nearly 40000 digits, it will be safe to allow 4N digits for each factorial. Thus, the loop that forms each new factorial can be terminated at an address that is moved 4 to the left for each new value of N. This scheme is a little wasteful (for example, factorial 1000 has 2568 digits, and this scheme is allowing 4000 digits), but on the side of discretion. And, of course, altering a test address by 4 each time around the main loop requires very little code.

N	First 40 digits of Factorial N	Number of digits	Number of zeros
1000	40238 72600 77093 77354 37024 33923 00398 57193	2568	249
2000	33162 75092 45063 32411 75393 38057 63240 38281	5736	499
3000	41493 59603 43785 40855 56867 09308 66121 70951	9131	748
4000	18288 01951 51406 50133 14743 17557 39190 44217	12674	999
5000	42285 77926 60554 35222 01064 20023 35844 05390	16326	1249
7000	88420 07956 96311 22478 64993 69689 77265 15146	23878	1749
10000	28462 59680 91705 45188 06413 21211 98688 90148	35660	2499
11000	31624 62406 47804 77296 47178 34667 73314 83123	39681	2748
12000	12018 5	43742	2998

A table of check values for calculating very large factorials. Thus, factorial 12000 has 43742 digits, of which the first few are 120185; it has 2998 low-order zeros. The low-order zeros can be counted precisely, as indicated in the text. The other figures for factorial 12000 are obtained by logarithms.

At that, the errors in the above scheme can be corrected if the whole procedure is halted every 1000 stages to record results and compare with previous values.

For example, suppose a HALT is arranged at factorial 1000, to check the leading digits with Croy's value (which, incidentally, was reproduced explicitly in our issue number 3). With the machine halted, it is a simple matter to alter the test address (which is at this point in error by over 1400 words) and restart the procedure, heading now for factorial 2000, at which point another correction can be applied.

When N is small, at the start of the calculation, this process runs at around four values of N per second on a 6502 microprocessor. At the other end, when N is approaching 11000, it runs around 36 seconds per N value. The total CPU time for factorial 11000 was 42 hours. It is estimated that nearly 39 billion instructions were executed.

The final result is a world's record. It is not, as the saying goes, "a pivotal event in world history," but it is a thoroughly satisfying job. Incidentally, in one sense, the result is the largest number ever calculated.

There was no attempt made in writing the program for factorial 11000 to conserve storage space or CPU time. With only modest effort, it would be feasible to pack two decimal digits in each 6502 word, this saving storage space (indeed, the 6502 permits decimal arithmetic directly for such an arrangement).

And consider those 2700 trailing zeros. From time to time during the calculation (say, every 1000 stages), all the trailing zeros could be shifted off. There must be better schemes for performing the multiplications, and more efficient ways to adjust each multiplication to its proper length. All the calculations can be done on any microprocessor.

So: it should not be too long before we have

factorial 12000.

Fred Gruenberger

21000	2.4601424077127746601930680919167321892214674707755	E4388
21001	3.9805940328442708593709727434659287838598635338190	E4388
22000	2.3911151843088049549559913282124240747696450497332	E4597
22001	3.8689056392276156629585369762006831399069933070766	E4597
23000	2.3240247421073882344725981460988351323814307931793	E4806
23001	3.7603510234254630904943010197956912774746730841490	E4806
24000	2.2588167384703365217149620777389398007434225675117	E5015
24001	3.6548422572021866571497016143087546277222699075298	E5015
25000	2.1954383555173030127807919148417209228490152223021	E5224
25001	3.5522938794321715091272953991088875073660950670711	E5224
30000	1.9042435673462438748500976847175750289440229160233	E6269
30001	3.0811308148245719945239489361992874536309118063108	E6269
32000	1.7988833295329268907179513697771116475942007181369	E6687
32001	2.9106543689798532156386630055228339095228058376828	E6687
35000	1.6516717741888653698044243887868187319557992250645	E7314
35001	2.6724610688964254538889073933075284541230358593124	E7314
37000	1.5602861269417554502388446439886711254341580619593	E7732
37001	2.5245959855666933431894037594510956684328822691348	E7732
41000	1.3924039510599758657318397124468407733649856320764	E8568
41001	2.2529569188846861267334287057293487307128165089770	E8568
45000	1.2425854011325228510806747464957412884780780084664	E9404
45001	2.0105454129568440559987031347177219784853851756110	E9404

Selected terms of the Fibonacci sequence,

showing the first 50 significant digits

and the proper power of 10.

..Fibonacci..

A much simpler high precision problem is that of calculating large terms of the Fibonacci sequence:

1, 1, 2, 3, 5, 8, 13, 21, 34,...

Tables of pairs of terms in this sequence have appeared in issues 25, 30, and 34, going as high as terms 20000 and 20001, at which point we were dealing with numbers of 4180 digits.

The scheme outlined in Flowcharts U and V is simple and straightforward. As with the factorial calculation, no attempt was made to conserve either storage space or CPU time.

The Fibonacci sequence gains one decimal digit about every 4.78 terms. Again, rather than devising an elaborate algorithm for controlling the number of digits being operated on, the simple rule was adopted of increasing the field length of each number by one digit every 4 terms.

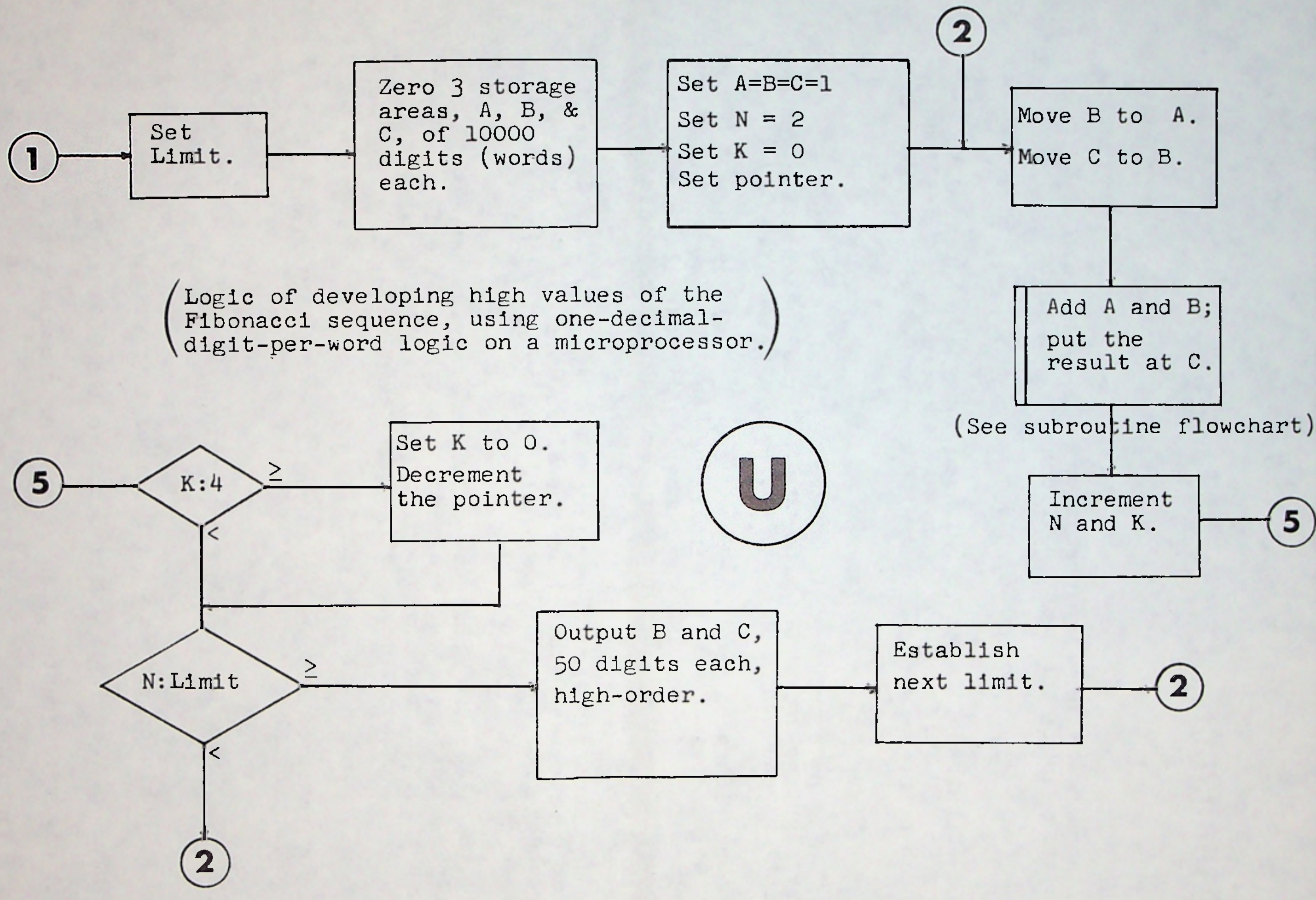
The accompanying table gives some new results, which are truncated from the exact explicit values; that is, they are NOT rounded from the 51st digit (as they were in the table in issue 34).

With access to 48K bytes of storage, even the one-digit-per-word scheme could extend this table quite far. For the effort of packing two digits per byte, a truly giant step could be taken in extending this table.

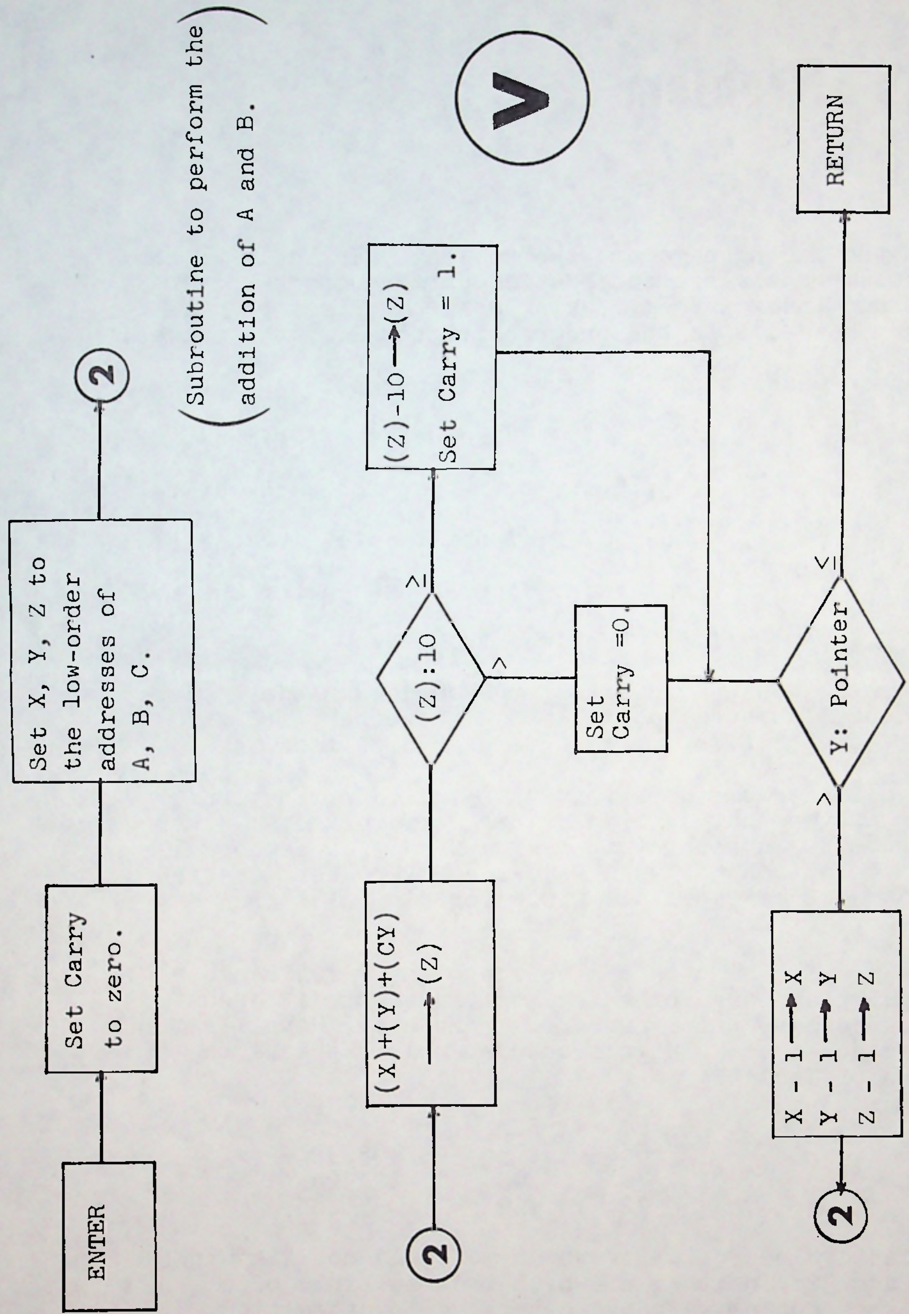
The ratio of successive terms, F, of the Fibonacci sequence approaches:

$$\tau = \frac{1+\sqrt{5}}{2}$$

Herman P. Robinson writes "...the error in τ obtained by dividing F_{40001} by F_{40000} will occur in the 16719th decimal. If that digit is greater than 6, the error could carry over to the 16718th decimal."



(Logic of developing high values of the Fibonacci sequence, using one-decimal-digit-per-word logic on a microprocessor.)



ENTER
Set Carry to zero.
Set X, Y, Z to the low-order addresses of A, B, C.
2
Subroutine to perform the addition of A and B.
V
2
(X)+(Y)+(CY) → (Z)
(Z):10
≥
(Z)-10 → (Z)
Set Carry = 1.
<
Set Carry =0.
Y: Pointer
>
≤
RETURN
X - 1 → X
Y - 1 → Y
Z - 1 → Z
2

Givoli's Problem

On the facing page are two tables. In Table W, the middle column shows the consecutive prime numbers. The first column indexes them, taking 2 as the first prime. The column headed S is the progressive total of the primes.

Nahman Givoli, of Tel Aviv, introduces this problem:

> If (p_i) is the sequence of primes, 2, 3, 5, 7, ..., then for each natural number d, find the smallest number N(d) such that d divides the sum of the first N primes (p_i).

Table Y shows the start of the desired table. Each of the natural numbers, d, is listed, together with the number, N, of the first prime for which d divides S.

Givoli raises a question about this distribution: Can it be proved that an N(d) exists for every d? There is no reason to suppose otherwise, and yet for some small values of d the respective N values are very hard to find. This appears to be especially true for d's which are multiples of 6.

The mention of 6 in connection with primes rings a bell, since 6 appears to be the most frequent difference between successive prime numbers. However, Givoli's conjecture does not hold good consistently; we have these extensions of Table Y:

60	103	25800
66	175	83292
70	209	122920
72	119	35568

and the last values of d for which we could not find an N were 303 and 326, neither of which is a multiple of 6. However, the problem warrants extensive investigation.

I	P	S	d	N	S
1	2	2	2	1	2
2	3	5	3	10	129
3	5	10	4	5	28
4	7	17	5	2	5
5	11	28	6	57	6870
6	13	41	7	5	28
7	17	58	8	11	160
8	19	77	9	20	639
9	23	100	10	3	10
10	29	129	11	8	77
11	31	160	12	97	22548
12	37	197	13	49	4888
13	41	238	14	5	28
14	43	281	15	57	6870
15	47	328	16	11	160
16	53	381	17	4	17
17	59	440	18	113	31734
18	61	501	19	23	874
19	67	568	20	9	100
20	71	639	21	40	3087
21	73	712	22	17	440
22	79	791	23	23	874
23	83	874	24	99	23592

W

Y

Three dice are tossed. The sum of the uppermost spots will be a number between 3 and 18. Out of 216 tosses of three dice, the sums have the expected distribution as follows:

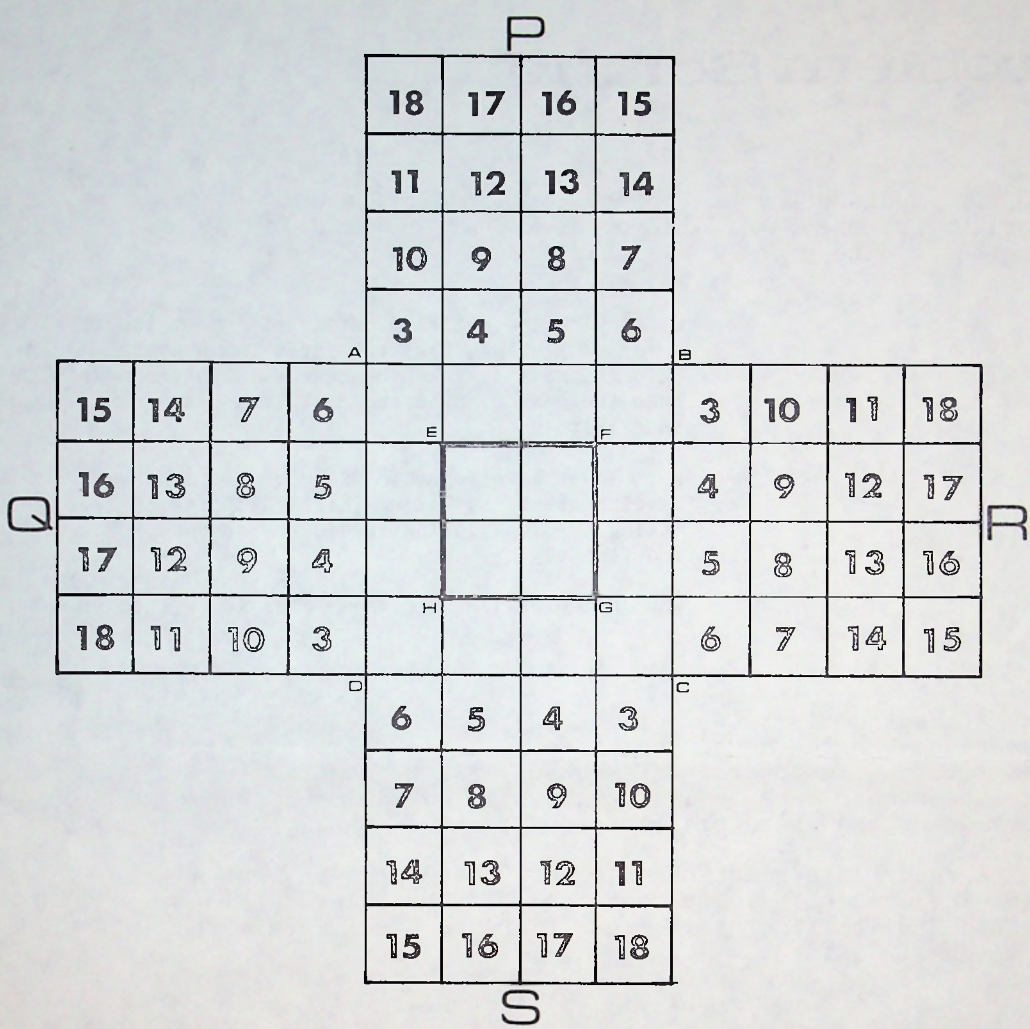
3	1	11	27
4	3	12	25
5	6	13	21
6	10	14	15
7	15	15	10
8	21	16	6
9	25	17	3
10	27	18	1

Four tosses are made of the three dice, to locate one cell in each of the four arrays P, Q, R, and S. For the selected cells, the centers from P to S are connected by a straight line, and similarly for the selected cells in Q and R.

These two straight lines intersect within square ABCD and possibly within square EFGH. The intersection could lie exactly on the border of the inner square EFGH, and in that case is to be discarded.

For example, if the dice call for the selection of cell 16 in array P, 18 in S, 3 in Q, and 11 in R, then the intersection of the two lines will fall on the border of the inner square.

The Problem, then, is: in the long run, what fraction of the intersections will fall inside EFGH?



Exploring Random Behavior--5

So you think you understand how random processes will behave?

Problem Solution

Dr. Rudi Borth, of Toronto, offers a different algorithm for the TAKE/SKIP7 problem that appeared in issue 88. The problem was this:

> Start with the positive integers. For the first level of sieving, Take the first two numbers, Skip the next 3, Take the next 4, Skip the next 5, Take the next 6, Skip the next 7, and so on, indefinitely.
>
> Each lower level operates in much the same way. Level K starts by Taking (K+1), Skipping (K+2), Taking (K+3), Skipping (K+4), and so on, indefinitely.
>
> What numbers will survive all levels of this sieve?

What follows is from Dr. Borth's letter:

After the first two terms (1,2), each level K = L adds one number to the series; namely, the number in position P = K + 1. To determine its value, all we need is to trace the number back to its position in level L = 1 (the natural numbers) where position equals value.

A recurrence relation which computes position Q of a number in level (L - 1)Sfrom its position P in level L is obtained from the fact that the difference Q - P equals the sum of the I "Skips":

$$(L+1) + (L+3) + (L+5) + \cdots + (L+2I-1) = IL + I^2$$

which have produced level L from level L-1. Thus, Q = P + I(L+I).

The required I is determined from the fact that I is also the number of "Takes" which have created level L; that is, the greatest possible number of terms in the sum:

$$L + (L+2) + (L+4) + \cdots + (L + 2(I - 1)) = I(L+I-1)$$

such that the sum does not exceed position P - 1. If I' is the positive root satisfying the second-degree equation

$$I'(L + I' -1) = P - 1$$

then the required I = INT(I').

An implementation of this algorithm on an H-P 9825A took 11 seconds to produce the first 30 members of the series. It seems that the time for each cycle is proportional to K rather than 2^K.

x--x--x--x--x--x--x--x--x--x--x--x--x--x

Dr. Borth is, of course, quite correct, and his solution to the problem is delightful. The presentation that appeared in issue 88 can be justified on these grounds, however:

(1) The overall objective was to show the great difference in execution time between floating point BASIC and machine language, using the identical algorithm in both situations. For this purpose, it is irrelevant how inefficient the solution is.

(2) The suggested problem offered another opportunity to demonstrate the "bucket brigade" method.

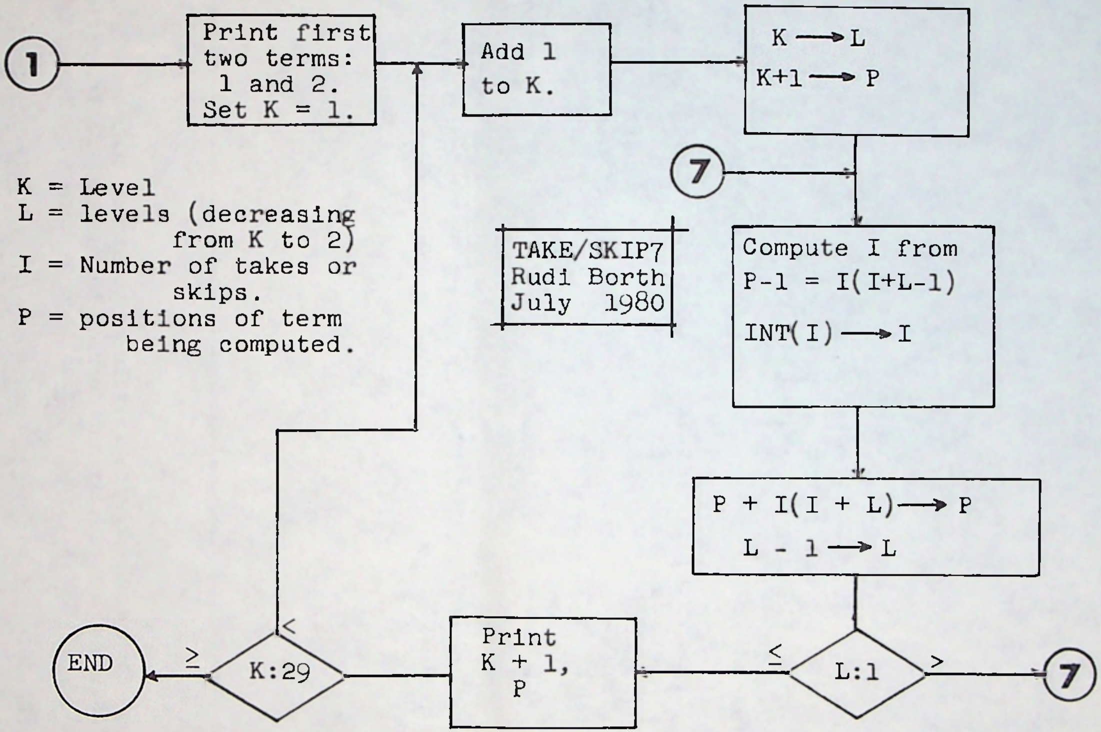
(3) We can't all devise brilliant and clever solutions; most of us just plug along. Thank heaven our computers are so fast.

A flowchart for Borth's algorithm is given on page 20.

John W. Wrench has written to point out that the result given in issue 89, page 20, is not correct beyond the 23rd significant digit. A better result (calculated by Herman P. Robinson) is:

8.70003 66252 08194 50322 24098 59113 00497 11932 97949 74289 20921 ...

Mr. Wrench also points out that proper terminology calls for labelling this result the limit of R_n as n approaches infinity. Finally, the last result given on page 20 of issue 89 seems to be exactly half its correct size, and should be 8.69866568.


1
Print first two terms: 1 and 2. Set K = 1.
Add 1 to K.
K → L
K+1 → P
7
Compute I from P-1 = I(I+L-1)
INT(I) → I
P + I(I + L) → P
L - 1 → L
L:1
>
≤
7
Print K + 1, P
K:29
<
≥
END
TAKE/SKIP7
Rudi Borth
July 1980
K = Level
L = levels (decreasing from K to 2)
I = Number of takes or skips.
P = positions of term being computed.